Remarks:

Claims 1, 3, 5, 7-24 and 26-28 stand rejected. Claim 23 is amended. Claims 1, 3, 5, 7-24 and 26-28 remain in the application.

ARGUMENT

Objections to the Specification:

The Examiner asserts that that amendment to paragraph [0018] introduces new matter not previously in the specification. However, it should be noted that Figure 1, as originally filed, shows **Applications 152** within the VM 15. Figure 4, similarly, show Applications 152 as part of the translated code region 155. It will be understood by those of skill in the art that a virtual machine (VM) on a platform may execute both an independent operating system (OS) in the guest VM, as well as applications executing under the OS. Figure 1 shows both that simulated OS 151 and Applications 152. It will be understood that the appearance of the block 152 shows that other applications may be executed on the VM 15. While there is no requirement or preference to running other applications, it will be understood that they may be executing in the VM. The appearance of **Applications 152** in Figure 1 obviates the new matter rejection. Thus, this rejection should be withdrawn.

§ 112 Rejections:

Claims 1, 3, 5, 7-24 and 26-28 are rejected under 35 U.S.C. § 112, second paragraph, as indefinite for failing to point out and distinctly claim the subject matter which applicant regards as the invention.

Regarding Claims 1, 9, 16 and 23, the Examiner asserts that it is unclear how translated code can reference a memory access. Claim 1, specifically, recited "*the monitor modifying original values of segment information in a descriptor table to force intervention by the monitor when the translated code references a memory access.*" Paragraph [0005], in the Background Section describes that "the instructions that access the memory or IO devices still require an address transformation." It will be understood by one of skill in the art that the phrase "instructions that access the memory or IO devices" is the same as saying "code that references a

memory access." When an instruction, or code, requires a memory access to fully execute, it "references" the memory access within the instruction. The Background Section further describes that "[o]n the other hand, a key condition for achieving good performance in ISA simulation is the efficient simulation of memory accesses, efficient address transformation scheme. In optimal case, the simulated CPU should be able to access the simulated memory without performing any address transformations." In at least one embodiment, the claimed invention is meant to solve this issue of address transformations. It will be understood that the addresses to be transformed, are "memory access" references within an instruction.

Paragraphs [0022-0023] describe that:

> "A sensitive instruction is translated into a sequence of pre-determined, simple instructions, called a "capsule". Simple sensitive instructions are emulated completely in the VM 15. Complex sensitive instructions are translated to a capsule that causes an exit from the VM 15 to the VMM 16. The VMM 16, after receiving the control, invokes an auxiliary ISA simulator 162 to simulate the original complex sensitive instruction. The linear address space of the VM 15 is constructed in a way that closely resembles the address space intended by the simulated OS 151. Thus, the instructions that access memory can do so natively using the original address. The only exception is the region in the upper part of the VM 15 linear address space, where the VMM 16 locates the translated code. This region is called a "translated code region" 155.
>
> When the VM 15 obtains the execution control back, the VM executes the translated code. Since the highest addresses of VM's 15 linear address space are occupied by the translated code, memory access by the directly executed instructions to this region should be forbidden. Such memory access is intercepted by using a segmentation mechanism."

This reference to "instructions that access memory" will be understood as "code that references a memory access," as recited in Claim 1; the terms "instructions" and "code" may be used interchangeably. Upon digesting the description in the Specification, one of skill in the art will understand that code that references a memory access must be specially treated upon simulation of the ISA to avoid natively executing instructions that will disturb certain areas of memory. Thus, simulation of the instructions takes care to treat code instructions that access memory, i.e., "*references a memory access*," specially. The claims explicitly recite that the monitor must modify the original values of the segment information in the descriptor table when the code has a memory access reference to prevent the translated code from being accessed or modified, directly.

The VM executes the translated code, but code that referenced a memory access must still reference some sort of memory access once it has been translated, or the original meaning of the code will be lost. One of skill in the art will understand this distinction. It is the form of the memory access in the translated code that may cause an exception, or interception by the monitor, when required.

Claims 1, 9, 16 and 23 were previously amended to more clearly recite which translated instructions are to be executed by which simulation module. As shown in Figure 1, embodiments of the invention may use three distinct simulation modules: (1) simulated OS **152**; (2) auxiliary simulator **162**; and platform device simulator **12**. This embodiment is clearly shown in Figure 1, and supporting description as originally filed, illustrating a virtual machine 15 with translated code region 155 and a host environment 101 executing a host OS 11 and platform simulator 12.

As is clearly described in the Specification, execution of the translated code depends on what type of instruction it is, on the target processor when an instruction require a device access, it is to be executed in the host environment using the full platform simulator. If the instruction is a "sensitive" instruction, as defined in the specification, it is to be executed by the VMM in the direct execution environment. Other instructions may be executed directly by the simulated OS in the VM, in the direct execution environment. The VMM controls the execution of the instructions based on the type of exception, if any, generated by an attempt of the VM to execute the translated code. These exceptions may be caused by various memory accesses in the translated instruction. Using the VMM and kernel to control execution of the translated code by capturing the exceptions allows the host machine to simulate execution of a first instruction set architecture even though the host machine utilizes a second instruction set architecture.

Only the instructions that do not compromise the integrity of the host OS are allowed to run natively. Instructions that access the privileged system state are intercepted and emulated in either the VMM simulator or the full platform simulator, for device access. A very efficient memory management scheme (based on segment virtualization) is put in place in order to allow instructions that access memory to run natively (with minimal management overhead).

§ 101 Rejections:

The Examiner asserts that Claim 23 appears to be directed to a system that is entirely software. Applicants point out that a "platform" as recited in the claim is a hardware platform, and that the Examiner's assertion is incorrect. In order to expedite allowance of the Claims, Applicants amend Claim 23 to clarify that the system has a hardware element by explicitly reciting that the platform includes a host processor to simulate the instruction set architecture of a target processor and that the virtual machine is running on the host processor on the platform. Thus, the amendment obviates this statutory subject matter rejection, and the rejection should be withdrawn.

CONCLUSION

In view of the foregoing, Claims 1, 3, 5, 7-24 and 26-28 are all in condition for allowance. If the Examiner has any questions, the Examiner is invited to contact the undersigned at (703) 633-6845. Early issuance of Notice of Allowance is respectfully requested. Please charge any shortage of fees in connection with the filing of this paper, including extension of time fees, to Deposit Account 50-0221 and please credit any excess fees to such account.

Respectfully submitted,

Dated: 31 Oct. 2008

/ Joni D. Stutman-Horn /
Joni D. Stutman-Horn, Reg. No. 42,173
Patent Attorney
Intel Corporation
(703) 633-6845

Intel Corporation
c/o Intellevate, LLC
P.O. Box 52050
Minneapolis, MN 55402